MASSACHUSETTS INSTITUTE OF TECHNOLOGY

A.I. LABORATORY

Artificial Intelligence
Memo No. 343

December 4, 1975
LOGO MEMO 20

Leading a Child to a Computer Culture

by

Cynthia J. Solomon*

*The research described in this paper was conducted at Massachusetts Institute of Technology in the Artificial Intelligence Laboratory's LOGO GROUP. Under the support of the National Institute of Education Grant No. NIE-G-74-0012. Paper to be presented at SIGCSE-SIGCUE Symposium February, 1976, Anaheim, California, Anaheim.

Leading a Child to a Computer Culture

by

Cynthia J. Solomon*

"LOGO" is sometimes used as the name of a programming language. It is also used as the name of ... what shall I call it? ... an environment, a culture, a way of thinking about computers and about learning and about putting the two together. I shall try to convey to you how I bring a child into this environment. The environment is made of ideas, of things and of people. The things include various types of turtles: computer controlled mechanical beasts which use touch sensors or eyesight to crawl around the floor and display turtles, which live on TV-like screens where they draw in phosphor white or in multi-color. The computer system which gives life to all of this understands the LOGO language. Some of the LOGO words are commands for the turtle, others are for the computer. There are commands and operations for one turtle type and not another. The set of operations developed in LOGO like XCOR and YCOR describe the display turtle's position while FTOUCH and RTOUCH give information about the floor turtle's state.

The flavor of this first paragraph already indicates a strong anthropomorphization of all of the components in this mini-world. This kind of representation is very much part of the cultural environment the kids will enter. I use an anthropomorphic "meta-language" in talking to the kids about computation.

The question arises whether this culture is closely tied to turtles. This is not my intention. I see it as a universal computer culture. The turtles were invented as vehicles to convey this culture to beginners. I believe that they make certain images more vivid and certain ideas more concrete. But the goal is to convey these images and ideas. To make them <u>real</u>, <u>comfortable</u>, <u>personal</u> for a beginner of any age. The turtle is a means to this end; so if you do not have turtles you should still be capable of connecting my experiences with your own.

Another aspect of the culture is learning to see projects as research enterprises. When I start with a child I try to convey to her that we are embarking on a research effort. We are trying to understand the turtle's behavior. To do this we might have to study our own behavior in certain situations. For example if we want to understand what we have to tell the turtle so that it can draw a square or a circle we will use ourselves as a model. We will stand up and walk in a square or circle and try to observe our own actions. We "play turtle." Our first attempt at "playing turtle" might feel confusing and need teacher feedback, but soon it will become an important problem solving tool. The first and hardest thing to come to grips with is the fact that the turtle's state is changed by either telling it to go FORWARD some amount or telling it to turn RIGHT some amount. These are separate functions always, not just when we want them to be. Children eventually see that they themselves

*The author is currently a student at Boston University. The research described in this paper was conducted at Massachusetts Institute of Technology in the Artificial Intelligence Laboratory's LOGO Group under the support of NSF and NIE.

The author wishes to thank Seymour Papert for his help in the development of the ideas in this paper. The author would also like to thank William Hennemen for his support and encouragement.

combine FORWARD and RIGHT into SIDESTEP but that this shortcutting is made up of understanding position and direction.

Playing turtle is an important aid in debugging, which itself is another key idea in this computer culture. "Which way should we turn the turtle?" "Stand up and be the turtle. Which way would you turn?" A big difficulty for a beginner is to realize that when the turtle or a person faces you and raises what looks like the same hand as you it/she is really raising the other one. It's the "mirror image" problem. (If you like you can call this the issue of relative coordinates, but I don't.) From our experience looking in mirrors we arrive at a "wrong" interpretation. A curiosity. But in the LOGO world this is an interesting bug worth thinking about. In fact you can hardly help doing so. Usually it is easy to fix. It will happen often, so we add it to our list of common bugs.

The idea of a bug collection is an exceedingly powerful component of the LOGO world. Some other worlds (like motorcycle maintenance) have trouble shooting check lists. That's a little like bug collections. But still very different. For one thing the bug collection is collected, not found in a book. For another we (always the child and I) can laugh together at funny bugs, at how some keep coming up, how we find some hard to fix. And in the computer world (I mean in our computer world though perhaps not IBM's image or even Dijkstra's) bugs and debugging are part of life, not occasional accidents, not a sort of plague. We live with them and learn to like living with them.

So when a child asks "How do you make the turtle do ..." One response might be "play turtle." Another suggestion, which also has important consequences, is "try something, whatever you 'feel like.' If you don't like what the turtle does you can 'undo it.'"

See how two great ideas come together. People are sometimes afraid of the explicit literal minded formal thinking of computer work. But in our environment it is not like that. Or rather it has a "follow your hunch" side and a "literal minded" side. The philosophy of happy debugging allows the two to co-exist. And kids can learn to see the two as styles to adopt for a purpose. So each is enriched by association with the other. And by dissociation from it.

One of the researchable questions for the teacher is what is discoverable and what information is better given to the child. It is essential for the child that she feel like an experimenter. She must not be timid about trying things out. While trying them she is looking for buggy situations. Some children are more resistent than others to this attitude. Sometimes this is because the child has not yet picked up enough of the "bug culture." There is quite a lot to pick up. For example contrast it with other cultures. We often see bugs as rather good things because we can learn from them. Other people see everything as either "right" or "wrong." For them, if it has a bug it is wrong and bad. But for us this might make it interesting. There are many ways to react to a buggy situation. "Can you recognize the bug?" "Is it a new one?" "Is it worth putting in our collection?" We learn to appreciate some bugs. They are telling us something. Of course, we also learn that some situations are buggier than others. And some so buggy that you might as well ERASE ALL.

The aspect of bugs I want to emphasize here is learning to recognize and appreciate them .. like we learn to recognize and appreciate people and kinds of people. More anthropomorphism! Anthropomorphism is central in all rich computer cultures. As for recognizing the hopelessly buggy situation, this is all about learning to make decisions about time. You can always debug it but sometimes it is not worth the time. So we ought to talk a little about decision making.

I want the kid to make the decisions. But sometimes she needs help and so I have the problem of how to intervene. What I want to show you is how my knowledge about kids and computers enters the decision guidance. This is a complicated process so I'll have to switch modes from general talk to a concrete story. It was time to do so anyway.

Let's look at a beginning child's first experience in a LOGO environment. (By the way, though I am talking explicitly about elementary

school children, most of what I say here is applicable to adults who have been deprived of this kind of computer culture.) In this example we pick a student somewhere between 6 and 10 years old who starts with the display turtle and a standard keyboard. The screen is bordered by red, green, blue and yellow strips of tape marked NORTH, EAST, SOUTH and WEST. We refer to either the name or color in initial discussions about where the turtle is headed.

My opening remarks would be along the following lines: The turtle is represented by the triangle on the screen. Notice its nose is pointing NORTH. We can tell it to move by typing FORWARD followed by some number. When we want the turtle to do it we press the DOIT button. (On some terminals it is called RETURN or CAR RET --if I can I'll paste DOIT on that key. This nice little metaphor is due to R. Perlman and her Button Box.) Now you tell the turtle to do something. By the way you can type FD instead of FORWARD. The turtle understands.

The child will probably type

```
FD 7
```

If you ask for a bigger number, the child will type 9. If they have never used a typewriter before children will ask how to make bigger numbers. They need to be told about concatenation. (A rather interesting simple fact! They know how to make CAT but not 37.)

Now, I'd say, if you want the turtle to face directly EAST tell it to turn RIGHT 90. I say "90 is a magic number." The children will try this. And then I'd suggest they teach the turtle to make a square or a box.

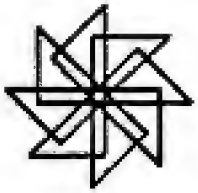
Here the possible bugs will be following through on using RIGHT 90 at each corner. How do you make the turtle head SOUTH from EAST, and so on. Another bug--a kind of local/global conflict, is to remember what the goal is and what special qualities distinguish squares from other objects. The angles and sides are the same. But often in constructing objects alone and for the first time little steps are taken. The lengths of the sides are eyeballed and therefore not quite the same. A lot of fudging takes place but isn't recognized as that by the child. Anyway the job becomes buggy and difficult. Here are 2 possible results:

(1a)

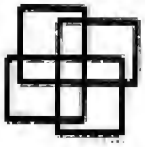
(1b)

One teaching technique might be to say that that's very nice. It's not quite a square but it's interesting. Let's give this drawing a name and teach it to the computer. In other words we'll follow through on this project.

After teaching it, use it with the child. Run it a few times. You may want to change the turtle's heading after running it each time. Most likely the pattern produced by multiple runnings of this procedure will be very pretty. (There is always a chance that it won't be.) This is a good example of looking for ways to capitalize on bugs. You can rely on the POLY theorem (the total turtle trip) to bring the turtle through interesting patterns if its stopping state is different from its starting state. Later this is important to call attention to. You, as teacher, should understand and be prepared for very interesting effects. Indeed, this kind of richness is a central (seldom appreciated) part of what makes turtle work so great for kids and others!

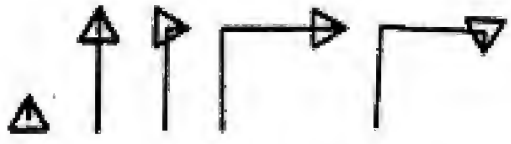


(2a) After running (1a) 8 times.



(2b) After running (1b) 4 times.

So we explore. But most often we eventually go back to the original project of building a square. Now is the time to discuss global strategy. What about a square. Use pencil and paper. Drew a square. Trace out the turtle's activities on paper. Like this



START FD 53 RT 90 FD 53 RT 90

FD 53 RT 90 FD 53

Now the child might again have difficulty. She might ask "How do you tell the turtle to go down?" She might get confused about heading the turtle south. Encourage the child to play turtle again. Ask her to notice whether she changes direction. When she commands the actual turtle and chooses 3 LEFT's instead of 1 RIGHT, that's fine. The difficulty is resolvable. It is a useful cliche.

She might have to be reminded that the command for moving the turtle from one location to another is FD. Remember it moves the turtle in the direction its nose is pointing.

Now we have understood how to drive the turtle in a square. The next step is to turn it into a named procedure. In the LOGO environment with small children my meta-language for this is: Teach the computer how to do it by itself. "TEACH" is built into my version of the LOGO operating system so that the word is used as the "definitional syntax." Part of TEACHing is giving the procedure a name (e.g., "SQUARE") so it can be used later as a sub-procedure for more complex projects. For the young kids this is seen as: the computer has learned a new word, SQUARE.

So much for this example. It's hardly a big programming project. That will soon come in the LOGO environment. Next week the child will be making movies on the screen. But though a square might be simple the experience of making it was not simple at all. Nor was it seen as simple or boring by the kid. And the skill and preparation and knowledge needed by the teacher was the least simple of all these things.

There are other papers which discuss more complex projects. I do not want to do that here. Instead I want to end on this question: What should a computer teacher know? My answer is that the computer teacher should have a computer culture. The LOGO culture is one possible one. I think it is vastly better than the BASIC or FORTRAN cultures. You will object that BASIC and FORTRAN are just languages. LOGO too. You are right. But there is a bug in what you say because around the languages there have developed cultures, or ways of thinking about computers, people, and learning. I don't want to knock anyone's language or culture. What I want to say takes the form of an appeal to be more explicit about computer cultures and environments. Let's worry more about how to make them richer and about how to bring more people, teachers, kids, anyone, into them. Especially let's encourage one another to talk about our computer culture(s). That's what I've tried to do here in a groping way. That's what I try to do when I introduce future LOGO teachers to the environment. With time the community of computer teachers will become more skilled at talking about these things. When that happens the integration of computers into education will have come of age.

REFERENCES

Abelson, Hal, et al., LOGO Manual, MIT AI Lab. LOGO Memo 7, 1973.

Goldberg, Adele and Bonnie Tenenbaum, Classroom Communication Media, TOPICS in Instructional Computing, vol. 1, Sigcue, Jan. 1976.

Inhelder, Barbel and Jean Piaget, The Early Growth of Logic in the Child, Norton, 1969.

Inhelder, Barbel, Hermine Sinclair, and Magali Bovet, Learning and the Development of Cognition, Harvard Univ. Press, 1974.

Kay, Alan. A Personal Computer for Children of All Ages, Proc. ACM National Conf., August 1972, Boston.

Minsky, Marvin and Seymour Papert, Artificial Intelligence, Oregon University Press, 1974. Also as AI Progress Rept., Mass. Inst. Tech., Artificial Intelligence Lab., Memo 252, 1972.

Minsky, Marvin, A Framework for Representing Knowledge, MIT, AI Memo 306, 1974.

Minsky, Marvin, Form and Content in Computer Science, JACM, vol. 17, no. 2, 1970. Also as MIT AI Memo 187, 1969.

Papert, Seymour and Cynthia Solomon, Twenty Things to Do with a Computer, MIT, AI Lab. LOGO Memo 3, July 1971. Also in EDUCATIONAL TECHNOLOGY, April 1972.

Papert, Seymour, Teaching Children to be Mathematicians vs. Teaching about Mathematics, MIT AI Lab. LOGO Memo 4, July, 1971. Also in Int. J. Math. Educ. Sci. Technol., vol. 3, 249-262, 1972.

Papert, Seymour, Uses of Technology to Enhance Education, MIT AI Lab. LOGO Memo 8, June 1973.

Papert, Seymour, On Making a Theorem for a Child, Proc. ACM Annual Conf. Aug. 1972. Also in NEW EDUCATIONAL TECHNOLOGY, General Turtle Development Inc., Cambridge, Ma.

Papert, Seymour and Cynthia Solomon, NIM: A Game Playing Program, MIT AI Lab. LOGO Memo 5, Feb. 1972.

Papert, Seymour, Teaching Children Thinking, MIT AI Lab. LOGO Memo 2, Oct. 1972.

Papert, Seymour, 5 Lectures in Process Models for Psychology, Rotterdam Univ. Press, The Netherlands, 1973.

Papert, Seymour, Curriculum Units, MIT AI Lab. LOGO Working Paper 4, Dec. 1972.

Papert, Seymour, LOGO Manual, MIT AI Lab. LOGO Working Paper 20, 1972 (revised).

Perlman, Radia, TORTIS--Toddler's Own Recursive Turtle Interpreter System, MIT AI Lab. LOGO Memo 9, Mar. 1974.

Piaget, Jean, Child's Conception of Number, Norton, 1965.

Solomon, Cynthia, Dividing the Swan, MIT AI Lab. LOGO Working Paper 14, 1973.

Solomon, Cynthia, Comments and Advice on Introducing Kids to Turtles and LOGO, MIT AI Lab. LOGO Working Paper 42, 1975.

Solomon, Cynthia (with S. Papert), Teaching an Ex-First Grader, MIT AI Lab. LOGO Working Paper 44, 1975.

Solomon, Cynthia and Seymour Papert, Teach: A Step Toward More Interactive Programming, MIT AI Lab. LOGO Working Paper 43, 1975.